# APL Problem Solving Competition
# Phase 1

## Introduction

The Phase 1 problems are designed to be solved using short APL functions. If you find yourself writing more than a couple of statements in your solution, then there is probably a better way to do it.

### Submission format

Each solution must be a single dfn or tacit function.

A dfn is one or more APL statements enclosed in braces `{}`. The left hand argument, if any, is represented in a dfn by `⍺`, while the right hand argument is represented by `⍵`. For example:

```
      'Hello' {⍺,'-',⍵,'!'} 'world'
Hello-world!
```

A dfn terminates on the first statement that is not an assignment. If that statement produces a value, then the dfn returns that value as its result. The diamond symbol `⋄` separates APL statements. For example:

```
      'left' { ⍵ ⋄ ⍺ } 'right'
right
```

More information on dfns can be found on the APL Wiki.

A tacit function is an APL expression that does not explicitly mention its arguments. In the example below, `(+⌿÷≢)` is a tacit function that computes the average of a vector (list) of numbers:

```
      (+⌿÷≢) 1 2 3 4 5 6
3.5
```

More information on tacit functions can be found on the APL Wiki.

### Judging Guidelines

Phase 1 will mainly be judged based on:

- Generality: does your function handle the given edge-cases?
- Use of array-oriented thinking: did you write array-oriented APL or something that looks more like C# written in APL?

You should not include comments in your Phase 1 solutions.

### Tips

- Several of the problem descriptions will describe arguments that can be a scalar (a single element) or a vector (a list). This is largely pedantic, but in such cases your functions should produce correct results for both types of input.
- The symbol `⍝` is the APL comment symbol. In some of the examples, we provide comments to give you more information about the problem.
- Some of the problem test cases use "boxed display" to make the structure of the returned results clearer. Boxing is always active on TryAPL and can be enabled in your local APL Session with the `]Box` user command:

```
      ⍳¨⍳4
 1   1 2   1 2 3   1 2 3 4
      ]Box on
Was OFF
      ⍳¨⍳4
```

| 1 | 1 2 | 1 2 3 | 1 2 3 4 |
|---|---|---|---|

# Sample problem

The content of the orange box shows what a typical Phase I problem description looks like. It also presents some possible solutions of varying quality, and explains how to provide your own solution.

Each problem starts with a task description; some also include a hint suggesting one or more APL primitives. These may be helpful in solving the problem, but you are under no obligation to use them. Clicking on a primitive in the hint opens the Dyalog documentation page for that primitive.

Each problem ends with some example cases. You can use these as a basis for implementing your solution.

## Counting Vowels A

Write an APL function to count the number of vowels (A, E, I, O, U) in an array consisting of uppercase letters (A–Z).

💡 **Hint:** The membership function `X∊Y` could be helpful for this problem.

### Examples

```
      (fn) 'COOLAPL'
3
      (fn) ''   ⍝ empty argument
0
      (fn) 'NVWLSHR'   ⍝ no vowels here
0
```

Below are three sample solutions. All three produce the correct answer, but the first two functions would be ranked higher by the competition judging committee as they demonstrate better use of array-oriented programming than the third one.

```
      ({+/⍵∊'AEIOU'}) 'COOLAPL'   ⍝ good dfn
3
      (+/∊∘'AEIOU') 'COOLAPL'   ⍝ good tacit function
3
      ⍝ suboptimal dfn:
      {(+/⍵='A')+(+/⍵='E')+(+/⍵='I')+(+/⍵='O')+(+/⍵='U')} 'COOLAPL'
3
```

# 1: Are You a Bacteria?

A DNA string is composed of the letters 'A','C','G' and 'T'. The GC-content of a DNA string is given by the percentage of the symbols in the string that are either 'C' or 'G'. Very small discrepancies in GC-content can be used to distinguish species; for instance, most bacteria have a GC-content significantly higher than 50%.

Write a function that:

- has a right argument that is a non-empty character vector representing a DNA string.
- returns the percentage of GC-content in the string.

💡 **Hint:** The *membership* function `X∊Y` could be helpful for this problem.

---

### Examples

```
      (fn) 'GCGCGCGCCCGGGGCCG'
100

      (fn) 'ACGTACGTACGTACGT'
50

      (fn) 10 12 16 10/'ACGT'
58.33333333
```

# 2: Index-Of Modified 🔍

Write a function that behaves like the APL *index-of* function `R←X⍳Y` except that it returns `0` instead of `1+≢X` for elements of `Y` not found in `X`.

---

### Examples

```
      'DYALOG' (fn) 'APL'
3 0 4

      (5 5⍴⎕A) (fn) ↑'UVWXY' 'FGHIJ' 'XYZZY'
5 2 0
```

# 3: Multiplicity ✖

Write a function that:

- has a right argument `Y` which is an integer vector or scalar
- has a left argument `X` which is also an integer vector or scalar
- finds which elements of `Y` are multiples of each element of `X` and returns them as a vector (in the order of `X`) of vectors (in the order of `Y`).

💡 **Hint:** The *residue* function `X|Y` and *outer product* operator `X∘.fY` might be useful for this problem.

---

### Examples

```
      ⎕←Y←20?20 ⍝ your example may be different
5 7 8 1 12 10 20 16 11 4 2 15 3 18 14 19 13 9 17 6

      2 4 7 3 9 (fn) Y ⍝ using ]Box on
```

| 8 12 10 20 16 4 2 18 14 6 | 8 12 20 16 4 | 7 14 | 12 15 3 18 9 6 | 18 9 |
|---|---|---|---|---|

```
      3 (fn) ⍳10
```

| 3 6 9 |
|---|

```
      6 7 (fn) 42
```

| 42 | 42 |
|---|---|

```
      2 3 5 (fn) ⍬ ⍝ returns a vector of 3 empty vectors
```

| | | |
|---|---|---|

```
      ⍬ (fn) ⍳10 ⍝ returns an empty vector
```

# 4: Square Peg, Round Hole

Write a function that:

- takes a right argument which is an array of positive numbers representing circle diameters
- returns a numeric array of the same shape as the right argument representing the difference between the areas of the circles and the areas of the largest squares that can be inscribed within each circle

💡 **Hint:** The *pi times* function `○Y` could be helpful.

---

## Examples

```
      (fn) 2×⍳5
1.141592654 4.566370614 10.27433388 18.26548246 28.53981634

      (fn) (2*.5)×3 3 ⍴⍳9
 0.5707963268  2.283185307  5.137166941
 9.132741229  14.26990817  20.54866776
27.96902001   36.53096491  46.23450247
```

# 5: Rect-ify

Suppose you have a number of trees that you want to plant in a rectangular pattern with complete rows and columns, meaning all rows have the same number of trees. You also want that rectangular pattern to be as "square" as possible, meaning there is a minimal difference between the number of rows and columns in the pattern.

Write a function that:

- has a right argument `N` which is a positive integer less than or equal to 1,000,000.
- returns a 2-element integer vector `R` representing the rows and columns of the rectangle such that:
  - `N=×/R` meaning `N` equals the number of rows × the number of columns (you planted all the trees!)
  - `≤/R` meaning the number of rows is less than or equal to the number of columns
  - `|-/R` is minimal, meaning the difference between the elements of `R` is as small as possible

---

## Examples using ]Box on

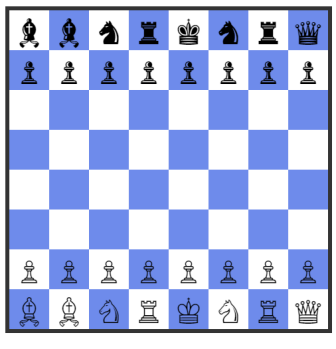
```
      (fn) 12
3 4

      (fn) 16
4 4

      (fn)¨⍳19
```

| 1 1 | 1 2 | 1 3 | 2 2 | 1 5 | 2 3 | 1 7 | 2 4 | 3 3 | 2 5 | 1 11 | 3 4 | 1 13 | 2 7 | 3 5 | 4 4 | 1 17 | 3 6 | 1 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

```
      (fn)¨999999 1000000
```

| 999 1001 | 1000 1000 |
|---|---|

# 6: Fischer Random Chess ♖



According to Wikipedia, Fischer random chess is a variation of the game of chess invented by former world chess champion Bobby Fischer. Fischer random chess employs the same board and pieces as standard chess, but the starting position of the non-pawn pieces on the players' home ranks is randomized, following certain rules. White's non-pawn pieces are placed on the first rank according to the following rules:

- The Bishops must be placed on opposite-color squares.
- The King must be placed on a square between the rooks.

The good news is that you don't actually need to know anything about chess to solve this problem! We'll use strings whose elements are `'KQRRBBNN'` for the King (♔), Queen (♕), 2 Rooks (♖), 2 Bishops (♗), and 2 kNights (♘) respectively.

Write a function that:

- has a character vector right argument that is a permutation of `'KQRRBBNN'`
- returns 1 if the following are true:
  - the K is between the two Rs
  - the Bs occupy one odd and one even position

  otherwise a 0 is returned.

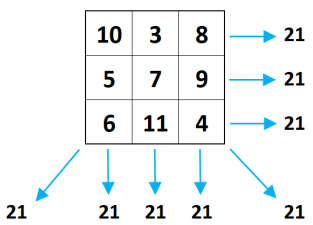
💡 **Hint:** The *where* function `⍸Y` and the *residue* function `X|Y` could help with solving this problem.

---

## Examples

```
      (fn) 'RNBQKBNR' ⍝ standard chess layout
1
      (fn) 'BBNRKNRQ' ⍝ layout in diagram above
1
      (fn) 'RBBNQNRK' ⍝ K not between Rs
0
      (fn) 'BRBKRNQN' ⍝ Bs both in odd positions
0
```

# 7: Can You Feel the Magic? 🪄




Wikipedia states that, in recreational mathematics, a square array of numbers, usually positive integers, is called a magic square if the sums of the numbers in each row, each column, and both main diagonals are the same.

Write a function to test whether an array is a magic square. The function must:

- have a right argument that is a square matrix of integers (not necessarily all positive integers)
- return 1 if the array represents a magic square, otherwise return 0

💡 **Hint:** The *dyadic transpose* `X⍉Y` function could be helpful for solving this problem.

---

## Examples

```
      (fn) 1 1⍴42
1
      (fn) 3 3⍴4 9 2 3 5 7 8 1 6
1
      (fn) 2 2⍴1 2 3 4
0
```

# 8: Time to Make a Difference ⏱️

Write a function that:

- has a right argument that is a numeric scalar or vector of length up to 3, representing a number of [[[days] hours] minutes] – a single number represents minutes, a 2-element vector represents hours and minutes, and a 3-element vector represents days, hours, and minutes.
- has a similar left argument, although not necessarily the same length as the right argument.
- returns a single number representing the magnitude of the difference in minutes between the arguments.

💡 **Hint:** The functions *decode* `X⊥Y` and *take* `↑` could be useful for this problem.

---

## Examples

```
      2 30 (fn) 5 15
165
      5 15 (fn) 2 30
165
      1 0 0 (fn) 0 ⍝ number of minutes in a day
1440
      1 0 0 (fn) ⍬ ⍝ don't forget to handle empty arguments!
1440
      1 0 (fn)¯1 0
120
      1.5 0 (fn) 90
0
```

# 9: In the Long Run 🏃

Write a function that:

- has a right argument that is a numeric vector of 2 or more elements representing daily prices of a stock.
- returns an integer singleton that represents the highest number of consecutive days where the price increased, decreased, or remained the same, relative to the previous day.

💡 **Hint:** The *N-wise reduction* operator `X f/ Y` function could be useful when solving this problem.

---

## Examples (the longest runs are highlighted)

```
      (fn) 1 2 3 5 5 5 6 4 3
3
      (fn) 1 2 3 4 4 4 4 4 5 4 3
4
      (fn) 1 2
1
```

# 10: On the Right Side ☰

Write a function that:

- has a right argument `T` that is a character scalar, vector or a vector of character vectors/scalars.
- has a left argument `W` that is a positive integer specifying the width of the result.
- returns a right-aligned character array `R←` of shape `((2=|≡T)/≢T),W` meaning `R` is one of the following:
  - a `W`-wide vector if `T` is a simple vector or scalar.
  - a `W`-wide matrix with the same number rows as elements of `T` if `T` is a vector of vectors/scalars.
- if an element of `T` has length greater than `W`, truncate it after `W` characters.

💡 **Hint:** Your solution might make use of *take* `X ↑ Y`.

---

## Examples

In these examples, `⍴⍞←` is inserted to display first the result and then its shape.

```
      ⍴⍞←6 (fn) '⍣'
     ⍣
6

      ⍴⍞←8 (fn) 'K' 'E' 'Iverson'
       K
       E
 Iverson
3 8

      ⍴⍞←10 (fn) 'Parade'
    Parade
10

      ⍴⍞←8 (fn) 'Longer Phrase' 'APL' 'Parade'
Longer P
     APL
  Parade
3 8

      starsForSpaces←'*'@(=∘' ')
      starsForSpaces 6 (fn) '⍣'
*****⍣
```